UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/752,796	12/29/2000	Adi Yoaz	42390P9574	9366

7590 05/26/2005

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP
Seventh Floor
12400 Wilshire Boulevard
Los Angeles, CA 90025

EXAMINER	
COLEMAN, ERIC	

ART UNIT	PAPER NUMBER
2183	

DATE MAILED: 05/26/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/752,796	YOAZ ET AL.	
	Examiner	Art Unit	
	Eric Coleman	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *02 March 2005*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-26* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-26* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

Claim Rejections - 35 USC § 103

1. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

2. Claims 1-18, are rejected under 35 U.S.C. 103(a) as being unpatentable over Kahle (patent No. 5,467,473) in view of Lipasti (On the Value Locality of Store Instructions) and Yoaz (patent No. 5,987,595).

3. Kahle taught the invention substantially as claimed including a data processing ("DP") system comprising:

a) Processing section (e.g., see fig. 1)

b) An extended load buffer (e.g., see fig. 5 that shows a layout of a load queue);

c) A marking section (e.g., see fig. 6 that shows in step 4 that a load program number is placed or marked into the load queue (extended load buffer). Therefore a marking unit must exist to perform marking).

d) A comparing processing section (e.g., see fig. 6 step 6 and col. 2, lines 56-59) show that a store address is compared to the load addresses to the load buffer. For this comparison to take place a comparison unit must exist); and

e) A recovery processing section (e.g., see fig. 6, steps 9 and 10 and col. 3, lines 4-11 that show that a load instruction must be placed in original order and re-executed.

It is shown that this is because of a conflict that exists and therefore original order must be recovered. This must be accomplished with the recovery unit).

4. Kahle taught advancing load instructions ahead of store instructions (e.g., see col. 2, lines 56-59 that shows when store executes, its address is compared to previously executed load instructions, in a load queue, which executed out of order ahead of the store. So in the time frame of the store execution the loads have been previously executed. However, in order to execute the load instructions out of order ahead of the store, the unexecuted load instructions are inherently advanced over the store for execution and once executed become previously executed loads that were previously advanced over the store. This is further shown in col. 3, lines 57-66. An example is in Figure 2 with col. 5, line 46-col. 6, line 13 where a load is executed ahead of a store instruction and thus the load is advanced ahead of a store instruction for execution (and thus was unexecuted when advanced). When the data being stored by the store instruction is the same as the data already in the register, the store clearly does not affect the execution a load instruction.

5. Kahle did not expressly detail a silent store instruction or advancing dependent loads ahead of silent stores the loads were dependent upon (claims 1,10). However Lipasti discusses the notion of a silent store on page 183 col. 2, last paragraph (where Lipasti defines a silent store as one that does not change the system state i.e., the value being written to by the store matches the exact value already stored at the memory location). Lipasti taught on page 185 col. 1, section 3.1, the concept of squashing of the silent stores. The motivation for squashing the store taught there was

for improving memory performance of the memory access path, and allowing a designer to obtain greater performance from existing structures or reduction in size and complexity of the system (wherein Lipasti refers to the load/store queue or LSQ). Clearly this is the same type of system taught by Kahle.

6. One of ordinary skill in the DP art would have been motivated to incorporate the Lipasti teachings of squashing of silent stores at least for the reasons taught by Lipasti that comprise improving memory performance of the memory access path, allowing a designer to obtain greater performance from existing structure or reduction in size and complexity of the system.

7. As to the implementation of squashing of silent stores Lipasti taught (on page 185-186 in section 3.4) each store was converted to a store verify that was effectively three operations a load, comparison, and subsequent store (if the store was non-silent). The store verify was initiated after the effective address had been computed in the execution engine and all previous store addresses were known, so that possible store address unknown dependencies need not be considered. When the data returned from the memory subsystem is was compared to the new value to be written. If the data values were equal, the store was update silent and was removed from the load/store queue and the store entry is flagged to indicate that the store is silent. When the store reached commit is was not flagged as silent the store port was obtained and the write to the memory subsystem occurred normally. If the store was silent the store retired with no memory access and no side effect, except that it consumed a commit slot. Lipasti further taught a perfect method for silent stored on page 186 col. 1, first full paragraph.

Lipasti taught that store squashing occurred in the same manner as described above except it is known by some mechanism that the store is silent and hence the verification is performed only for known silent stores. Non-silent stores execute as normal with no store verify and this method was taught as illustrating the performance obtained with a perfect prediction mechanism for update-silent stores. Lisasti acknowleged that no confidence mechanism can ever be perfect-hence validation of the prediction must still be done.

8. Kahle taught the advancing of loads past stores if the load and store did not access the same location and if the load was after the store in the queue so they would not conflict as determined by comparisions of program numbers (e.g., see col. 7, line col. 8, line 52 of Kahle). Lipasti taught squashing silent stores (with the same type of system as Kahle) for increasing performance providing a system that compares addresses to determine if a load and store access the same location, compare program instructions to determine if the load is after the store in program order. In the situation where there was a potential conflict because the load was after a store instruction and where both instructions accessed the same memory location but stores that would have stored the same value was already in the register or memory location would have been squashed. Since the silent stores would have been squashed then one of ordinary skill would have been motivated perform advancing of loads past silent stores even when a store would have accessed the same location because store would have been squashed or not effectively performed. Since Lipasti taught prediction of silent stores

(as discussed above) one of ordinary skill would have been motivated to use the prediction of silent store as an indication as to whether to advance loads past stores.

9. Kahle and Lipasti did not disclose (claims 1,10), a predictor having a collision history table (CHT). Yoaz however, taught a predictor having a collision history table (CHT) (e.g., see fig. 3, element 88)(e.g., see col. 3, lines 50-52) that show that the CHT was used for predicting and thus was part of a predictor along with the control unit (e.g., see fig.3, element 102). Figure 3 shows that the CHT or predictor was coupled to a recorder buffer, 94 using some control logic. Column 6, lines 35-36 show that this buffer holds entries for load instructions. Yoaz disclosed in col. 3, lines 50-60 that the predictor was used for predicting load instructions so that loads can be executed ahead of stores. Lipasti discusses the notion of a silent store on page 183, column 2.

10. Yoaz taught (e.g., see col. 2, lines 58-63 that his method was able to execute more load instructions out-of-order (based on the predictor for faster processor operations. These faster processor operations would have motivated one of ordinary skill in the art to modify the design of Kahle and Lipasti to use the collision history table and predictor as described by Yoaz. Page 185, section 3.1 of Lipasti shows squashing of silent stores (using prediction as shown in section 3.4 under the perfect method) allows a designer to obtain greater performance for existing structures or reduction in size or complexity of the system. This ability to obtain greater performance or reduction in size would have motivated one or ordinary skill in the art to implement the prediction as taught by Lipasti using the CHT as taught by Yoaz, that already was used for

predicting load instructions so load could be executed of load, for the prediction of silent stores.

11. It would have been obvious to one of ordinary skill in the DP art to combine the teachings of Kahle and Lipasti and Yoaz. One of ordinary skill would have motivated to modify the design of Kahle to include a predictor having a collision history table as disclosed by Yoaz, and predicting silient stores as taught by Lipasti so that the processor operations may be sped up and greater performance or smaller area may be realized. It would have been obvious to one of ordinary skill in the art to compare an unexecuted load instruction with an unexecuted store instruction, and have the unexecuted load instruction bypass the issued store instruction for execution if the unexecuted load instruction value and the issued and unexecuted store value are the same so that a memory access was avoided and performance was sped up.

12. As per claim 2,11, the predictor being a silent store predictor was obvious in view of the teachings of Kahle, Lipasti and Yoaz as discussed above.

13. As per claim 3,12 Kahle in view of Lipasti and Yoaz taught path based indexing and the path was based on branching (as discussed above). Yoaz. taught (e.g., see col. 5, lines 9-23) show that the sequence of instructions was based on correct prediction of branches. As shown in column 4, lines 8-10, the tag of the CHT is a linear instruction pointer. Thus, the predictor was indexed based on the linear sequence of instructions that is used based on branches.

14. As per claim 4,13 Kahle in view of Lipasti and Yoaz taught the system of claim 3 wherein the silent predictor was coupled with a state machine. Yoaz taught (e.g., see

col. 4, lines 43-45) to show that the CHT includes prediction bits being either sticky or saturating counters. A saturating counter in itself is a state machine because it varies its state based inputs.

15. A per claim 5,14, Kahle in view of Lipasti and Yoaz has disclosed the apparatus of claim 4 where the state machine is one of 1-bit, 2-bit, and a sticky bit. Yoaz taught (e.g., see col. 4, lines 43-45) that the CHT included prediction bits being either sticky or saturating counters.

16. As per claim 6,15 Kahle in view of Lipasti and Yoaz taught the system of claim 1, as described above, wherein the predictor is memory dependent. Column 3, lines 65-60 of Yoaz that shows that the predictor was based on memory addresses and thus was memory dependent.

17. As per claim 7,16 Kahle in view of Lipasti and Yoaz has disclosed the system of claim 1, wherein the extended load buffer comprises bit field to mark load address match, load data match, load predict, and load flush, and bit fields for load address, load attribute and load data. As shown in figure 5 the extended load buffer holds a load address. This address is updated as a result of a load instruction or an instruction that was matched as a load. Therefore, this field is also the load instruction or an instruction that was matched as a load. Therefore, this field is also the load address match field. Figure 5 also shows that the table includes a PC field, which gives the age of the instruction. This is load data of a load instruction that is also a load attribute. Since the data is written there upon realizing that an instruction matches a load instruction, the field is also a load data match. Column 9, lines 10-21 show that a load can be marked

upon a match indicating that the load must be re-executed. Since the extended load buffer holds information for the loads, it is clear that this buffer would then hold the marking bits in such an embodiment as the prediction bits. Since the marking bits set above are marked not only on a match of addresses but also on improper ordering, these bits also signify a load flush because the load and subsequent instructions need to be flushed for re-execution as shown previously.

18. As per claim 8,17 Kahle in view of Lipasti and Yoaz has disclosed the system of claim 1, as described above wherein the CHT is one of indexed by a tag and tagless. Yoaz has shown in figure 2A a tagged CHT.

19. As per claim 9, 18 Kahle in view of Lipasti and Yoaz had disclosed the apparatus of claim 1, as described above, wherein the CHT includes distance bits Yoaz has shown in figure 2D a CHT including distance bits.

20. As to the further limitations of claim 10 Kahle taught a processor (figure 1) having and internal memory (figure 1, element 1); Bus coupled to the processor (figure 1, element 2); Memory coupled to a memory controller and the processor (e.g., see col. 2, line 65-col. 3, line 1). It is inherent that the memory has control logic so that it can be manipulated. ; An extended load buffer (e.g., see figure 5 that gives a layout of a load queue that is also a buffer); A marking process (figure 6 shows in step 4 that a load program number is placed or marked into the load queue (extended load buffer). Therefore a marking unit must exist to perform this marking; A comparing process(figure 6, step 6 and column 2, lines 56-59) show that a store address is compared to the load addresses of the load buffer. For this comparison to take place a comparison

unit must exist; and A recovery process (figure 6, steps 9 and 10 and column 3, lines 4-11 show that a load instruction must be place in original order and re-executed. It is shown that this is because of a conflict that exists and therefore original order must be recovered. This must be accomplished with a recovery unit. Wherein the unexecuted load instructions are advanced over silent store instructions (this is discussed above).

21. Claims 19-20,22-24,26 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kahle (patent No. 5,467,473) in view of Lipasti (On the Value Locality of Store Instructions).

22. As to the further limitations of claim 19, Kahle taught a method comprising; fetching and instruction (figure 6, step 1) and determined if an instruction is one of a store and a load (figure 6, step 3); issuing the store instruction (figure 6, step 5 shows that shows are executed and thus must by issued in the pipeline). Comparing an addres and data of the store address with load instructions in an extended load buffer (figure, steps 6 and 8); Figure 5 gives a layout of the load queue used in figure 6 to hold load information. This queue is also a buffer. The PC value indicates a program a program number for comparison of age of the instructions and thus is data of the instructions; Setting marking bits in the extended load buffer if a match is found in the comparing (col. 9, lines 10-21 show that a load can be marked upon a match indicating that the load must be re-executed. Since the extended load buffer holds information for the load, it is clear that this buffer would then hold the marking bits in sub and embodiment. Updating a memory with store instruction if the store instruction can be retired; col. 6, lines 37-39 show that the memory is updated when the result of a store is committed of

retired. The bypassing of a store instruction and executing a load instruction ahead of the silent store instruction and performing a silent store prediction is discussed below.

23. Kahle taught advancing load instructions ahead of store instructions (e.g., see col. 2, lines 56-59 that shows when store executes, its address is compared to previously executed load instructions, in a load queue, which executed out of order ahead of the store. So in the time frame of the store execution the loads have been previously executed. However, in order to execute the load instructions out of order ahead of the store, the unexecuted load instructions are inherently advanced over the store for execution and once executed become previously executed loads that were previously advanced over the store. This is further shown in col. 3, lines 57-66. An example is in Figure 2 with col. 5, line 46-col. 6, line 13 where a load is executed ahead of a store instruction and thus the load is advanced ahead of a store instruction for execution (and thus was unexecuted when advanced). When the data being stored by the store instruction is the same as the data already in the register, the store clearly does not affect the execution a load instruction.

24. Kahle did not expressly detail a silent store instruction or advancing dependent loads ahead of stores the loads were dependent upon (claims 19,23). However Lipasti discusses the notion of a silent store on page 183 col. 2, last paragraph (where Lipasti defines a silent store as one that does not change the system state i.e., the value being written to by the store matches the exact value already stored at the memory location). Lipasti taught on page 185 col. 1, section 3.1, the concept of squashing of the silent stores. The motivation for squashing the store taught there was for improving memory

performance of the memory access path, and allowing a designer to obtain greater performance from existing structures or reduction in size and complexity of the system (wherein Lipasti refers to the load/store queue or LSQ). Clearly this is the same type of system taught by Kahle.

25. On of ordinary skill in the DP art would have been motivated to incorporate the Lipasti teachings of squashing of silent stores at least for the reasons taught by Lipasti that comprise improving memory performance of the memory access path, allowing a designer to obtain greater performance from existing structure or reduction in size and complexity of the system.

26. As to the implementation of squashing of silent stores Lipasti taught (on page 185-186 in section 3.4) each store was converted to a store verify that was effectively three operations a load, comparison, and subsequent store (if the store was non-silent). The store verify was initiatited after the effective address had been computed in the execution engine and all previous store addresses were known, so that possible store address unknown dependencies need not be considered. When the data returned from the memory subsystem is was compared to the new value to be written. If the data values were equal, the store was update silent and is was removed from the load/store queue and the store entry is flagged to indicate that the store is silent. When the store reached commit is was not flagged as silent the store port was obtained and the write to the memory subsystem occurred normally. If the store was silent the store retired with no memory access and no side effect, except that it consumed a commit slot. Lipasti further taught a perfect method for silent stored on page 186 col. 1, first full paragraph.

Lipasti taught that store squashing occurred in the same manner as described above except it is known by some mechanism that the store is silent and hence the verification is performed only for known silent stores. Non-silent store execute as normal with no store verify and this method was taught as illustrating the performance obtained with a perfect prediction mechanism for update-silent stores. Lisasti acknowleged that no confidence mechanism can ever be perfect-hence validation of the prediction must still be done.

27. The combination of the Kahle teachings comprised the advancing of loads past stores if the load and store did not access the same location and if the load was after the store in the queue so they would not conflict as determined by comparisions of program numbers (e.g., see col. 7, line col. 8, line 52 of Kahle). The Lipasti teachings of squashing silent stores with the same type of system for increasing performance provides a system that compares addresses to determine if a load and store access the same location, compare program instructions to determine if the load is after the store in program order. In the situation where there was a potential conflict because the load was after a store instruction and where both instructions accessed the same memory location but stores that would have stored the same value was already in the register or memory location would have been squashed. Since the silent stores would have been squashed then one of ordinary skill would have been motivated perform advancing of loads past silent stores even when a store would have accessed the same location because store would have been squashed or not effectively performed. Since Lipasti taught prediction of silent stores (as discussed above). One of ordinary skill would have

been motivated to use the prediction of silent store as an indication as to whether to advance loads past stores.

28. As per claim 20 Kahle in view of Lipasti has disclosed the method of claim 19, as described above, further comprising preparing the load instruction for retirement, if the load instruction is complete, and determining if the load instruction is marked flush in the extended load buffer. (column 6, lines 20-24 of Kahle show that a load is committed or retired. In Preparation for this the address and program number are removed from the load queue. Since the marking bits set above are marked not only on a match of addresses bus also on improper ordering, these bits also signify a load flush because the load and subsequent instructions need to be flushed for re-execution as shown previously.

29. As per claim 22 Kahle in view of Lipasti discloses the method of claim 19 wherein the memory is a cache. As show in column 6, lines 37-39 of Kahle, the completed store operation writes to a memory via a cache thus the operation writes t the cache memory as well as a memory.

30. As per claim 23, Kahle taught a program storage device readable by a machine comprising instructions that cause the machine to: fetch and operation (figure 6, step 1) and determining if an instruction is one of a store and a load instruction (figure 3, step 3); Execute the store instruction (figure 6, step 5); compare and address and data of the store operation with load operations in an extended load buffer (figure 6, steps 6 and 8); (figure 5 gives a layout of the load queue used in figure 6 to hold information. This queue is also a buffer. The PC value indicates a program number for comparison

of age of the instructions and thus is data of the instructions. Setting marking bits in the extended load buffer if a match is found in the compare instruction; column 9, lines 10-21 show that a load can be marked upon a match indicating that the load must be re-executed. Since the extended load buffer holds information for the loads, it is clear that this buffer would then hold the marking bits in such an embodiment. Update a memory with store operation if the store operation can be retired; (column 6, lines 37-39 show that the memory is updated when the result of a store is committed or retired.); As to the bypassing of silent store operation and executing a load operation ahead of a silent store operation and performing a silent store prediction if the operation is a store this is discussed above.

31. As per claim 24. Kahle in view of Lipsasti disclosed the method of claim 23 as described above, wherein the instructions further cause the machine to prepare the load operation for retirement if the load operation is complete, and determining to prepare the load operation for retirement if the load operation is complete, and determining if the load operation is marked flush in the extended load buffer (Column 6, lines 20-24 of Kahle show that a load is committed or retired. In preparation for this, the address and program number are remove from the load queue. Since the marking bits set above are marked not only on a match of addresses but also on improper ordering, these bits also signify a load flush because the load and subsequent instructions need to be flushed for rexecution as shown previously.

32. As per claim 26, Kahle in view of Lipasti taught the program storage device of claim 23, wherein the memory is a cache . (As shown in column 6, lines 37-39 of Kahle,

the completed store operation writes to a memory via a cache, thus the operation writes to the cache as well as a memory.)

33. Claims 21 and 25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kahle in view of Lipasti as applied to claims 19-20,22-24, and 26 above, and further in view of Yoaz.

34. As per claim 21, Kahle in view of Lipasti taught the method of claim 19 as shown above, Kahle in view of Lipasti did not teach wherein the predicting includes marking bits in a collision history table (CHT).

35. Yoaz taught predicting that includes marking bits in a collision history table (CHT) (figure 3, element 88). Column 3, lines 50-52 show that the CHT is used for predicting. Column 5, lines 57-67, show updating or marking the CHT. Yoaz has shown in column 2, lines 58-63 that his method is able to execute more load instructions out of order (based on the predictor for faster processor operation. These faster processor operations would have motivated one of ordinary skill in the art to modify the design of Kahle in view of Lipast to use the collision history table and predictor described by Yoaz.

36. As per claim 25, Kahle in view of Lipasti taught the method of claim 23 as shown above. Kahle in view of Lipasti did not teach wherein the instruction that causes the machine to predict silent stores includes and instruction that causes the machine to mark bits in a collision history table (CHT).

37. Yoaz taught wherein the instruction causes the machine to predict stores includes an instruction that causes the machine to mark bits in a collision history tale

(CHT) (figure 3, element 88). Column 3, lines 50-52 show that the CHT is used for predicting. Column 5, lines 57-67 show updating of marking the CHT. Since Lipasti taught the prediction of silent store one ordinary skill would have been motivated to incorporate the Yoaz method of using a CHT for predicting collision in the prediction of silent stores to take advantage of the improved efficiency of the Yoaz method of predicting.

38. Yoaz has shown in column 2, lines 58-63 that his method is able to execute more load instructions out of order (based on the predictor) for faster processor operations. These faster processor operations would have motivated one of ordinary skill in the art to modify the design of Kahle in view of Lipasti to use the collision history table and predictor described by Yoaz.

39. It would have been obvious to one of ordinary skill in the art at the time of invention to modify the design of Kahle in view of Lipasti to include the collision history table predictory disclosed by Yoaz so that the processor operations may be sped up.

Response to Amendment

Applicant's arguments with respect to claims 1-26 have been considered but are moot in view of the new ground(s) of rejection.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Eric Coleman whose telephone number is (571) 272-4163. The examiner can normally be reached on Monday-Thursday.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

EC

ERIC COLEMAN
PRIMARY EXAMINER